THE GALACTIC EMPIRE

A DATABASE DESIGN PROPOSAL

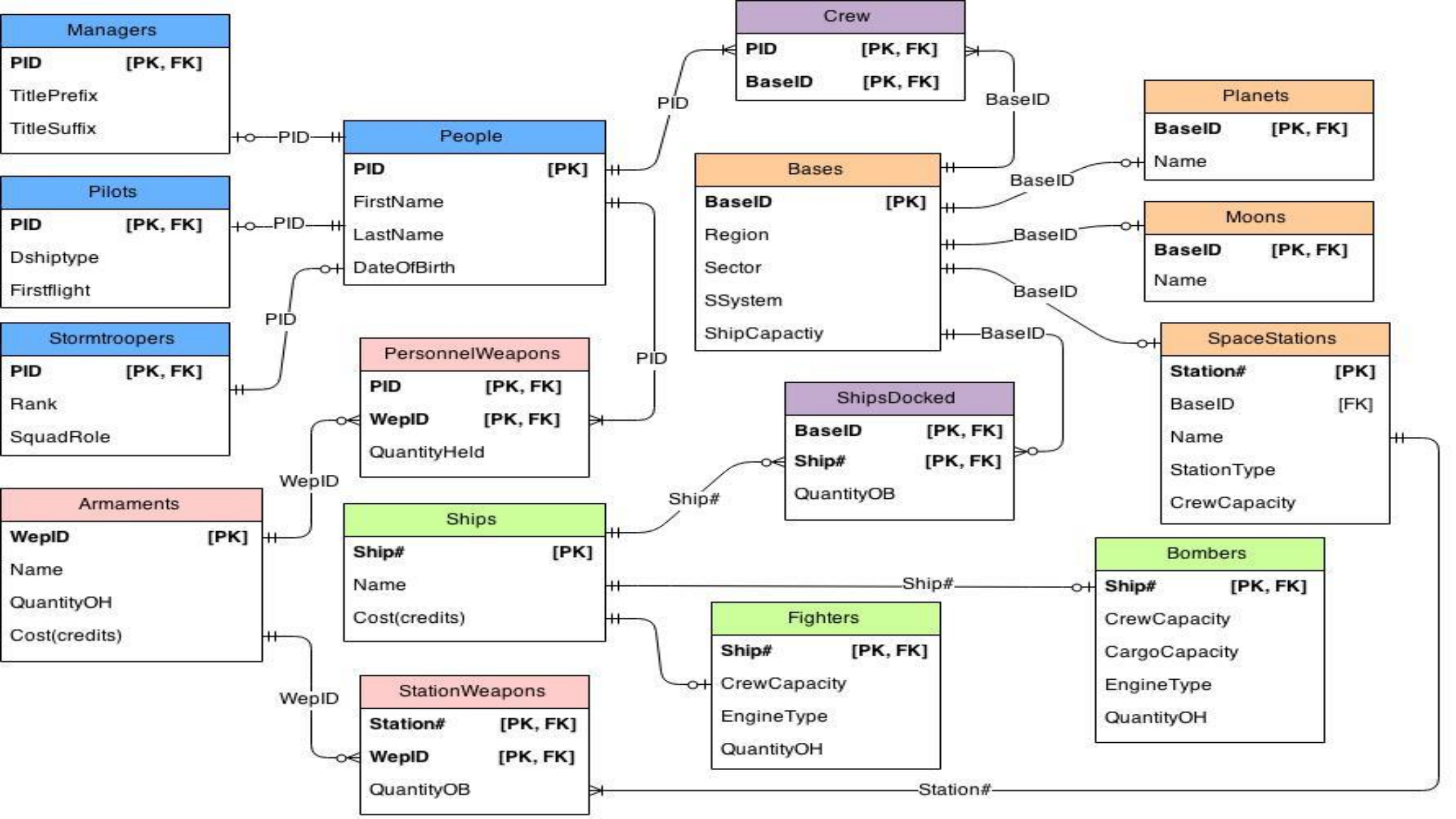
BY

TOM GONZALEZ

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This piece of writing presents a basic outline of the structure and entities involved in the implementation of a database system for the Galactic Empire. The purpose of this database is to enable to the tracking of equipment, that is, be able to find how much of what is where (and depending on the item, who has it). This database will allow the Galactic Empire to reduce its inventory costs by allowing it to track its items with greater accuracy. This database also is intended to allow for the tracking of employees meaning that an administrator should be able to determine an employees location using this database.


Managers
PID [PK, FK]
TitlePrefix
TitleSuffix
Pilots
PID [PK, FK]
Dshiptype
Firstflight
Stormtroopers
PID [PK, FK]
Rank
SquadRole
People
PID [PK]
FirstName
LastName
DateOfBirth
Crew
PID [PK, FK]
BaseID [PK, FK]
Bases
BaseID [PK]
Region
Sector
SSystem
ShipCapactiy
Planets
BaseID [PK, FK]
Name
Moons
BaseID [PK, FK]
Name
SpaceStations
Station# [PK]
BaseID [FK]
Name
StationType
CrewCapacity
PersonnelWeapons
PID [PK, FK]
WepID [PK, FK]
QuantityHeld
ShipsDocked
BaseID [PK, FK]
Ship# [PK, FK]
QuantityOB
Armaments
WepID [PK]
Name
QuantityOH
Cost(credits)
Ships
Ship# [PK]
Name
Cost(credits)
Bombers
Ship# [PK, FK]
CrewCapacity
CargoCapacity
EngineType
QuantityOH
Fighters
Ship# [PK, FK]
CrewCapacity
EngineType
QuantityOH
StationWeapons
Station# [PK, FK]
WepID [PK, FK]
QuantityOB
PID
BaseID
WepID
Ship#
Station#

PEOPLE LISTS ALL PEOPLE AND BASIC QUALITIES

```
CREATE TABLE people (
  pid                 char(6) not null unique,
  firstname           text not null,
  lastname            text not null,
  DateOfBirth         date not null,
 primary key(pid)
);
```

Functional Dependencies

pid -> firstname, lastname, DateOfBirth

	pid character(6)	firstname text	lastname text	dateofbirth date
1	p00001	Anakin	Skywalker	05/25/1977
2	p00002	Anson	Trask	01/08/1983
3	p00003	Boba	Fett	08/09/1991
4	p00004	Sheev	Palpatine	05/25/1977
5	p00005	Cassio	Tagge	07/15/1980
6	p00006	Trech	Molock	03/23/1974
7	p00007	Cassio	Tagge	07/15/1973
8	p00008	William	Touno	05/12/1969
9	p00009	Trel	Skutu	11/02/1989
10	p00010	Biggs	Darklighter	11/17/1970
11	p00011	Tycho	Celchu	03/09/1971
12	p00012	Maarek	Stele	10/04/1970
13	p00013	Juno	Eclipse	06/30/1972
14	p00014	CC	2224	11/17/1970
15	p00015	CT	7567	01/01/1971
16	p00016	CT	3110	01/01/1971
17	p00017	CT	3002	01/01/1971
18	p00018	Ran	Harkas	07/17/1981

STORMTROOPERS LISTS ALL STORMTROOPERS, THEIR RANK AND SQUAD ROLE

```
CREATE TABLE stormtroopers (
    pid          char(6) not null unique references  people(pid),
    rank         text not null,
    squadrole    text not null,
  primary key(pid)
);
```

Functional Dependencies

pid -> rank, squadrole

	pid character(6)	rank text	squadrole text
1	p00002	corporal	leader
2	p00014	commander	tactician
3	p00015	commander	engineer
4	p00016	Private1	sharpshooter
5	p00017	Private1	grenadier
6	p00018	sergeant	rifleman

PILOTS LISTS ALL PILOTS THE DATE OF THEIR FIRST FLIGHT, AND THEIR DESIGNATED SHIP TYPE.

```
CREATE TABLE pilots (
    pid          char(6) not null unique references people(pid),
    Dshiptype    text not null,
    firstflight  date not null,
  primary key(pid)
);
```

Functional Dependencies

pid -> Dshiptype, firstflight

	pid character(6)	dshiptype text	firstflight date
1	p00007	Bomber	09/03/1995
2	p00009	Fighter	04/26/2009
3	p00010	Fighter	12/25/1989
4	p00011	Bomber	02/14/1994
5	p00012	Fighter	03/01/2000
6	p00013	Fighter	06/30/1992

MANAGERS LISTS ALL MANAGERS AND THEIR OFFICIAL TITLES

```
CREATE TABLE managers (
    pid                 char(6) not null unique references people(pid),
    TitlePrefix         text not null,
    TitleSuffix         text not null,
  primary key(pid)
);
```

Functional Dependencies

pid -> TitlePrefix, TitleSuffix

	pid character(6)	titleprefix text	titlesuffix text
1	p00001	Darth	Vader
2	p00003	Bounty	Hunter
3	p00004	Galactic	Emperor
4	p00005	Imperial	General
5	p00006	Imperial	General
6	p00008	Imperial	General

BASES LISTS ALL IMPERIAL BASES, THE GALACTIC ADDRESS OF EACH BASE (BROKEN DOWN BY REGION, SECTOR AND SOLAR SYSTEM) AND HOW MANY SHIPS EACH BASE CAN HOLD.

```
CREATE TABLE Bases (
    BaseID              char(5) not null unique,
    Region              text not null,
    Sector              text not null,
    SSystem             text not null,
    ShipCapacity        integer not null,
  primary key(BaseID)
);
```

Functional Dependencies

BaseID -> Region, Sector, Ssystem, ShipCapacity

SAMPLE DATA ON FOLLOWING PAGE

	baseid character(5)	region text	sector text	ssystem text	shipcapacity integer
1	B0001	Deep Core	Sector 5	Prakith	20
2	B0002	Inner Rim	Adari Sector	Adim	100
3	B0003	Mid Rim	Msst Sector	Garos IV	225
4	B0004	Outer Rim Territories	Kanz Sector	Jerne	30
5	B0005	Galactic Core	Farlax Sector	Nzoth	125
6	B0006	Galactic Core	Corusca Sector	Coruscant	75
7	B0007	Expansion Region	Brak Sector	Genesia	20
8	B0008	Outer Rim Territories	Atrivis Sector	Horuz	20
9	B0009	Outer Rim Territories	Moddell Sector	Endor	40
10	B0010	Outer Rim Territories	Anoat Sector	Hoth	50

Planets lists all planets that have bases and the planet's name

```
CREATE TABLE Planets (
    BaseID      char(5) not null unique references Bases(BaseID),
    PName       text not null,
  primary key(BaseID)
);
```

Functional Dependencies

BaseID -> PName

	baseid character(5)	pname text
1	B0001	Prakith
2	B0002	Adim
3	B0003	Garos IV
4	B0004	Jerne
5	B0006	Coruscant
6	B0010	Hoth

Moons lists all moons that have bases and the moon's name

```
CREATE TABLE Moons (
    BaseID      char(5) not null unique references Bases(BaseID),
    MName       text not null,
  primary key(BaseID)
);
```

Functional Dependencies

BaseID -> MName

	baseid character(5)	mname text
1	B0007	Grimm
2	B0009	Endor

SpaceStations lists all spacestations, their name, their type, crew capacity and baseid.

```
CREATE TABLE SpaceStations (
    StationNum                  char(5) not null unique,
    SSName                      text not null,
    StationType                 text not null,
    CrewCapacity                integer not null,
    BaseID                      char(5) not null unique references Bases(BaseID),
  primary key(StationNum),
  foreign key(BaseID) references Bases(BaseID)
);
```

Functional Dependencies

StationNum -> SSName, StationType, CrewCapacity, BaseID

	stationnum character(5)	ssname text	stationtype text	crewcapacity integer	baseid character(5)
1	00015	Black Fifteen	Repair Yard	200	B0005
2	10000	The Death Star	Battle Station	12000	B0008

Ships lists all ships, their name and cost in credits

```
CREATE TABLE Ships (
    ShipNum          char(5) not null unique,
    ShipName         text not null,
    costCREDITS      integer not null,
  primary key(ShipNum)
);
```

Functional Dependencies

ShipNum -> ShipName, costCREDITS

	shipnum character(5)	shipname text	costcredits integer
1	s0001	TIE/LN starfighter	60000
2	s0002	TIE/SA bomber	65000
3	s0003	TIE/IN interceptor	72000
4	s0004	TIE/X1 advanced	90000
5	s0005	Scimitar assualt bomber	71500
6	s0006	Neutralizer-class bomber	83000
7	s0007	TIE/D automated fighter	170000
8	s0008	Starhunter	100000

Fighters lists all starfighters, their crew capacity, engine type, and quantity on hand

```
CREATE TABLE Fighters (
    ShipNum          char(5) not null unique references Ships(ShipNum),
    CrewCapacity     integer not null,
    EngineType       text not null,
    QuantityOH       integer not null,
  primary key(ShipNum)
);
```

Functional Dependencies

ShipNum -> CrewCapacity, EngineType, QuantityOH

	shipnum character(5)	crewcapacity integer	enginetype text	quantityoh integer
1	s0001	2	SFS P-s4 twin ion	212
2	s0003	1	SFS P-s5.6 twin ion	71
3	s0004	1	SFS P-s5.6 twin ion	70
4	s0007	0	SFS P-s4 twin ion	35
5	s0008	1	SFS P-s7.2 twin ion	35

Bombers lists all bombers, their crew and cargo capacity, engine type, and quantity on hand

```
CREATE TABLE Bombers (
   ShipNum              char(5) not null unique references Ships(ShipNum),
   CrewCapacity         integer not null,
   CargoCapTons         integer not null,
   EngineType           text not null,
   QuantityOH           integer not null,
 primary key(ShipNum)
);
```

	shipnum character(5)	crewcapacity integer	cargocaptons integer	enginetype text	quantityoh integer
1	s0002	1	15	SFS P-s4 twin ion	141
2	s0005	2	3	Single SFS P-s4 ion	35
3	s0006	1	2	single E-16/x ion	35

Functional Dependencies

ShipNum -> CrewCapacity, CargoCapTons, EngineType, QuantityOH

ShipsDocked lists all different ships, the bases they're docked at and how many ships are docked there

```
CREATE TABLE ShipsDocked (
   ShipNum              char(5) not null references Ships(ShipNum),
   BaseID               char(5) not null references Bases(BaseID),
   QuantityOB           integer not null,
 primary key(BaseID,ShipNum)
);
```

Functional Dependencies

(BaseID, ShipNum) -> QuantityOB

SAMPLE DATA ON FOLLOWING PAGE

	shipnum character(5)	baseid character(5)	quantityob integer
1	s0001	B0001	12
2	s0002	B0001	4
3	s0003	B0001	2
4	s0004	B0001	0
5	s0005	B0001	0
6	s0006	B0001	0
7	s0007	B0001	0
8	s0008	B0001	0
9	s0001	B0002	50
10	s0002	B0002	18
11	s0003	B0002	9
12	s0004	B0002	8
13	s0005	B0002	6
14	s0006	B0002	6
15	s0007	B0002	2
16	s0008	B0002	1
17	s0001	B0003	80
18	s0002	B0003	40
19	s0003	B0003	20
20	s0004	B0003	20
21	s0005	B0003	8
22	s0006	B0003	8
23	s0007	B0003	10
24	s0008	B0003	5
25	s0001	B0004	15
26	s0002	B0004	5
27	s0003	B0004	4
28	s0004	B0004	0
29	s0005	B0004	0
30	s0006	B0004	0
31	s0007	B0004	0
32	s0008	B0004	2
33	s0001	B0005	20
34	s0002	B0005	30
35	s0003	B0005	5
36	s0004	B0005	22
37	s0005	B0005	4
38	s0006	B0005	6
39	s0007	B0005	8
40	s0008	B0005	0

	shipnum character(5)	baseid character(5)	quantityob integer
41	s0001	B0006	16
42	s0002	B0006	11
43	s0003	B0006	20
44	s0004	B0006	11
45	s0005	B0006	5
46	s0006	B0006	3
47	s0007	B0006	3
48	s0008	B0006	5
49	s0001	B0007	4
50	s0002	B0007	9
51	s0003	B0007	2
52	s0004	B0007	0
53	s0005	B0007	0
54	s0006	B0007	0
55	s0007	B0007	0
56	s0008	B0007	5
57	s0001	B0008	2
58	s0002	B0008	3
59	s0003	B0008	0
60	s0004	B0008	2
61	s0005	B0008	4
62	s0006	B0008	3
63	s0007	B0008	2
64	s0008	B0008	4
65	s0001	B0009	3
66	s0002	B0009	5
67	s0003	B0009	4
68	s0004	B0009	5
69	s0005	B0009	6
70	s0006	B0009	8
71	s0007	B0009	7
72	s0008	B0009	2
73	s0001	B0010	10
74	s0002	B0010	16
75	s0003	B0010	5
76	s0004	B0010	2
77	s0005	B0010	2
78	s0006	B0010	1
79	s0007	B0010	3
80	s0008	B0010	11

BaseCrew lists which people are on which base

```
CREATE TABLE BaseCrew (
   pid                  char(6) not null unique references people(pid),
   BaseID               char(5) not null references Bases(BaseID),
  primary key(pid, BaseID)
);
```

Functional Dependencies

(pid, BaseID) -> N/A

	pid character(6)	baseid character(5)
1	p00001	B0008
2	p00002	B0010
3	p00003	B0009
4	p00004	B0008
5	p00005	B0005
6	p00006	B0004
7	p00007	B0007
8	p00008	B0001
9	p00009	B0002
10	p00010	B0009
11	p00011	B0003
12	p00012	B0002
13	p00013	B0004
14	p00014	B0008
15	p00015	B0008
16	p00016	B0006
17	p00017	B0008
18	p00018	B0010

Armaments lists weapons, weapon name, quantity on hand, and cost in credits

```
CREATE TABLE Armaments (
    WepID               char(4) not null unique,
    WName               text not null,
    QuantityOH          integer not null,
    CostCredits         integer not null,
  primary key(WepID)
);
```

Functional Dependencies

WepID -> WName, QuantityOH, CostCredits

	wepid character(4)	wname text	quantityoh integer	costcredits integer
1	w001	E-11 blaster rifle	100	1000
2	w002	SE-14r repeating blaster	100	400
3	w003	DLT-19 heavy blaster rifle	40	2000
4	w004	E-11s sniper rifle	20	1600
5	w005	DLT-20a blaster rifle	15	1300
6	w006	T-21 light repeating blaster	10	2000
7	w007	L-s1 laser cannon	100	10000
8	w008	Taim&Bak XX-9 heavy turbolasers	120	5000
9	w009	Borstel NK-7 ion cannon	120	1000
10	w010	interplanatary laser XS-1	1	5000000

PERSONNELWEAPONS LISTS WEAPONS, WHO HAS WHICH WEAPONS AND HOW MANY ARE THEY HOLDING

```
CREATE TABLE PersonnelWeapons (
    pid                 char(6) not null references people(pid),
    WepID               char(4) not null references Armaments (WepID),
    QuantityHeld        integer not null,
 primary key(pid, WepID)
);
```

Functional Dependencies

(pid, WepID) -> QuantityHeld

	pid character(6)	wepid character(4)	quantityheld integer
1	p00002	w001	1
2	p00002	w002	1
3	p00003	w001	1
4	p00003	w002	2
5	p00003	w006	1
6	p00005	w002	1
7	p00006	w002	1
8	p00007	w001	1
9	p00007	w002	1
10	p00008	w002	1
11	p00009	w002	1
12	p00010	w002	1
13	p00011	w002	1
14	p00012	w002	1
15	p00013	w002	1
16	p00014	w003	1
17	p00014	w002	1
18	p00015	w005	1
19	p00015	w001	1
20	p00015	w002	1
21	p00016	w004	1
22	p00016	w002	1
23	p00017	w003	1
24	p00017	w002	1
25	p00018	w005	1
26	p00018	w002	1

StationWeapons lists weapons, which station they are on, and how many are on board

```
CREATE TABLE StationWeapons (
    StationNum              char(5) not null references SpaceStations(StationNum),
    WepID                   char(4) not null references Armaments (WepID),
    QuantityOnBoard         integer not null,
  primary key(StationNum, WepID)
);
```

	stationnum character(5)	wepid character(4)	quantityonboard integer
1	00015	w007	60
2	00015	w008	40
3	00015	w009	50
4	10000	w007	40
5	10000	w008	80
6	10000	w009	70
7	10000	w010	1

VIEW PERSONNELLOCATION LISTS NAMES AND GALACTIC ADDRESS OF EACH PERSON

```
CREATE VIEW PersonnelLocation AS
    SELECT firstname, lastname, region, ssystem, sector
    FROM people, BaseCrew, Bases
    WHERE people.pid = BaseCrew.pid
    AND BaseCrew.BaseID = Bases.BaseID
    ORDER BY region;
```

	firstname text	lastname text	region text	ssystem text	sector text
1	William	Touno	Deep Core	Prakith	Sector 5
2	Cassio	Tagge	Expansion Region	Genesia	Brak Sector
3	CT	3110	Galactic Core	Coruscant	Corusca Sector
4	Cassio	Tagge	Galactic Core	Nzoth	Farlax Sector
5	Trel	Skutu	Inner Rim	Adim	Adari Sector
6	Maarek	Stele	Inner Rim	Adim	Adari Sector
7	Tycho	Celchu	Mid Rim	Garos IV	Msst Sector
8	CC	2224	Outer Rim Territories	Horuz	Atrivis Sector
9	CT	7567	Outer Rim Territories	Horuz	Atrivis Sector
10	CT	3002	Outer Rim Territories	Horuz	Atrivis Sector
11	Anakin	Skywalker	Outer Rim Territories	Horuz	Atrivis Sector
12	Ran	Harkas	Outer Rim Territories	Hoth	Anoat Sector
13	Anson	Trask	Outer Rim Territories	Hoth	Anoat Sector
14	Boba	Fett	Outer Rim Territories	Endor	Moddell Sector
15	Sheev	Palpatine	Outer Rim Territories	Horuz	Atrivis Sector
16	Trech	Molock	Outer Rim Territories	Jerne	Kanz Sector
17	Biggs	Darklighter	Outer Rim Territories	Endor	Moddell Sector
18	Juno	Eclipse	Outer Rim Territories	Jerne	Kanz Sector

VIEW PERSONNELWEAPONSINVENTORY LISTS WEAPON NAMES AND INVENTORY QUANTITY OF EACH WEAPON

```
CREATE VIEW PersonelIWeaponsInventory AS
    SELECT Wname, QuantityOH - sum(PersonnelWeapons.quantityheld) as InStock
    FROM Armaments, PersonnelWeapons
    WHERE Armaments.WepID = PersonnelWeapons.WepID
    GROUP BY armaments.Wname, QuantityOH, Armaments.WepID
    ORDER BY Armaments.WepID;
```

	wname text	instock bigint
1	E-11 blaster rifle	96
2	SE-14r repeating blaster	83
3	DLT-19 heavy blaster rifle	38
4	E-11s sniper rifle	19
5	DLT-20a blaster rifle	13
6	T-21 light repeating blaster	9

VIEW WeaponTracker lists people's names, the name of the weapon and how many they have

```
CREATE VIEW WeaponTracker AS
    SELECT firstname, lastname, WName, QuantityHeld
    FROM people
    INNER JOIN PersonnelWeapons
    ON people.pid = PersonnelWeapons.pid
    INNER JOIN Armaments
    ON PersonnelWeapons.WepID = Armaments.WepID;
```

	firstname text	lastname text	wname text	quantityheld integer
1	Anson	Trask	E-11 blaster rifle	1
2	Anson	Trask	SE-14r repeating blaster	1
3	Boba	Fett	E-11 blaster rifle	1
4	Boba	Fett	SE-14r repeating blaster	2
5	Boba	Fett	T-21 light repeating blaster	1
6	Cassio	Tagge	SE-14r repeating blaster	1
7	Trech	Molock	SE-14r repeating blaster	1
8	Cassio	Tagge	E-11 blaster rifle	1
9	Cassio	Tagge	SE-14r repeating blaster	1
10	William	Touno	SE-14r repeating blaster	1
11	Trel	Skutu	SE-14r repeating blaster	1
12	Biggs	Darklighter	SE-14r repeating blaster	1
13	Tycho	Celchu	SE-14r repeating blaster	1
14	Maarek	Stele	SE-14r repeating blaster	1
15	Juno	Eclipse	SE-14r repeating blaster	1
16	CC	2224	DLT-19 heavy blaster rifle	1
17	CC	2224	SE-14r repeating blaster	1
18	CT	7567	DLT-20a blaster rifle	1
19	CT	7567	E-11 blaster rifle	1
20	CT	7567	SE-14r repeating blaster	1
21	CT	3110	E-11s sniper rifle	1
22	CT	3110	SE-14r repeating blaster	1
23	CT	3002	DLT-19 heavy blaster rifle	1
24	CT	3002	SE-14r repeating blaster	1
25	Ran	Harkas	DLT-20a blaster rifle	1
26	Ran	Harkas	SE-14r repeating blaster	1

Reports Interesting Queries – queries that exemplify the potential utility of databases.

1. Query to return the number of stormtroopers within the same sector

```
SELECT sector, count(s.pid)
FROM stormtroopers s
INNER JOIN BaseCrew bc
ON s.pid = bc.pid
INNER JOIN bases b
ON bc.baseid = b.baseid
GROUP BY sector
```

	sector text	count bigint
1	Atrivis Sector	3
2	Anoat Sector	2
3	Corusca Sector	1

2. Query to return the number of each weapon held by stormtroopers

```
SELECT Wname, count(s.pid)
FROM PersonnelWeapons PW
INNER JOIN Armaments A
ON A.WepID = PW.WepID
INNER JOIN stormtroopers s
ON s.pid = PW.pid
group by a.wname
```

	wname text	count bigint
1	DLT-20a blaster rifle	2
2	SE-14r repeating blaster	6
3	DLT-19 heavy blaster rifle	2
4	E-11 blaster rifle	2
5	E-11s sniper rifle	1

3. Query to return total cost of ships docked (in credits)

```
SELECT sum(costCREDITS*QuantityOB)
from Ships, ShipsDocked
where Ships.ShipNum = ShipsDocked.ShipNum
```

	sum bigint
1	48154500

4. Query to return % of pilots who have been flying for more than 20 years

```
SELECT TRUNC (
        CAST(
            ( select count(pilots.pid)
              from people, pilots
              where people.pid = pilots.pid
              and (date_part('year', age(firstflight)))>20
             )as decimal
            )
            /
           ( select count(pilots.pid)
             from pilots
             )
          *100
        ) as Percent_Over20
```

	percent_over20 numeric
1	50

Stored Procedures

1. Function gives stormtroopers who are riflemen one EF-11 Blaster and one SE14-r repeating blaster

```
CREATE OR REPLACE FUNCTION add_PersonnelWeapons() RETURNS trigger AS
    $BODY$
        BEGIN
            IF NEW.squadrole = 'rifleman' THEN
                INSERT INTO PersonnelWeapons (pid, WepID, QuantityHeld)
                VALUES (NEW.pid, 'w001', 1);
                INSERT INTO PersonnelWeapons (pid, WepID, QuantityHeld)
                VALUES (NEW.pid, 'w002', 1);
            END IF;
        RETURN NEW;
        END;
    $BODY$
LANGUAGE plpgsql;
```

SAMPLE DATA FOR THIS PROCEDURE WILL BE PAIRED WITH THE SAMPLE DATA FOR THE TRIGGER THAT ACTIVATES IT IN THE FOLLOWING SECTION

Stored Procedures

2. Function returns galactic address given a BaseID

```
CREATE OR REPLACE FUNCTION BaseLocation (IN BaseID varchar(5))
RETURNS TABLE ("region" text,"sector" text,"ssystem" text) AS
$BODY$
        BEGIN
                RETURN QUERY SELECT Bases.region as region, Bases.sector as sector, Bases.ssystem as ss
                        FROM Bases
                        WHERE Bases.BaseID = BaseLocation.BaseID;
        END;
$BODY$
LANGUAGE plpgsql;
```

```
select BaseLocation('B0001')
```

	baselocation record
1	("Deep Core","Sector 5",Prakith)

TRIGGERS – CALL FUNCTIONS UPON THE OCCURRENCE OF A SPECIFIED ACTIVITY

1. Trigger

```
CREATE TRIGGER add_PersonnelWeapons
AFTER INSERT ON Stormtroopers
FOR EACH ROW
EXECUTE PROCEDURE add_PersonnelWeapons();
```

```
INSERT INTO people(pid, firstname, lastname, DateOfBirth)
VALUES ('p00019', 'Alan', 'Labouseur', '1970-07-15');

INSERT INTO stormtroopers(pid, rank, squadrole)
VALUES ('p00019', 'Private1', 'rifleman');
```

TRIGGERS

1. PersonnelWeapons Before Insert

	pid character(6)	wepid character(4)	quantityheld integer
1	p00002	w001	1
2	p00002	w002	1
3	p00003	w001	1
4	p00003	w002	2
5	p00003	w006	1
6	p00005	w002	1
7	p00006	w002	1
8	p00007	w001	1
9	p00007	w002	1
10	p00008	w002	1
11	p00009	w002	1
12	p00010	w002	1
13	p00011	w002	1
14	p00012	w002	1
15	p00013	w002	1
16	p00014	w003	1
17	p00014	w002	1
18	p00015	w005	1
19	p00015	w001	1
20	p00015	w002	1
21	p00016	w004	1
22	p00016	w002	1
23	p00017	w003	1
24	p00017	w002	1
25	p00018	w005	1
26	p00018	w002	1

1. PersonnelWeapons After Insert

	pid character(6)	wepid character(4)	quantityheld integer
1	p00002	w001	1
2	p00002	w002	1
3	p00003	w001	1
4	p00003	w002	2
5	p00003	w006	1
6	p00005	w002	1
7	p00006	w002	1
8	p00007	w001	1
9	p00007	w002	1
10	p00008	w002	1
11	p00009	w002	1
12	p00010	w002	1
13	p00011	w002	1
14	p00012	w002	1
15	p00013	w002	1
16	p00014	w003	1
17	p00014	w002	1
18	p00015	w005	1
19	p00015	w001	1
20	p00015	w002	1
21	p00016	w004	1
22	p00016	w002	1
23	p00017	w003	1
24	p00017	w002	1
25	p00018	w005	1
26	p00018	w002	1
27	p00019	w001	1
28	p00019	w002	1

Security

There are numerous potential users of this database but the current primary users are armory staff and managers. Managers are considered administrators and allowed to manipulate all while armory clerks are limited.

Admin

```
CREATE ROLE admin;
GRANT ALL ON ALL TABLES
IN SCHEMA PUBLIC
TO admin;
```

Armory Clerks

```
CREATE ROLE Armory;
GRANT SELECT ON ALL TABLES
IN SCHEMA PUBLIC
TO Armory;
GRANT INSERT ON Armaments
TO Armory;
GRANT UPDATE ON Armaments, PersonnelWeapons, StationWeapons
TO Armory;
```

NOTES-ISSUES-FUTURE PLANS

If I were to actually create a database that fully and accurately represented the equipment and personnel of the Galactic Empire according to various sources (*Star Wars* films, books, and Wikki) it would have taken an insurmountable amount of time. After going through all the steps to create this database I have realized a few errors I made which include: bases should not be split into sub-entities name is a common quality, quantity of ships on-hand should be a part of the ships table and not each sub-entity, and people cannot transfer bases. In the future I plan to correct these errors by modifying my entities: combining base sub- entities with bases, moving the ships on hand from each ship sub-entity to ship, adding arrival date to BaseCrew (this will also allow for calculations such as how long an individual has been on a certain base). I also would add more SpaceStations in the future but they were very difficult to determine the location of. The majority of my data entries are authentic pieces of information I found by search the website: http://starwars.wikia.com. In my next round of editing I also plan to include check constraints to ensure that my total number of ships at any give base is =< its ship capacity and that the total number of ships on bases is <= the total amount of ships on hand. I would also employ similar constraints on my armaments, PersonnelWeapons, and StationWeapons tables (making sure that the number of weapons held or on board are <= the quantity on-hand.